Remote visualization: ECho, cluster computing, and the Access Grid

M. D.Wolf,
K. Schwan, and G. Eisenhauer

Georgia Institute of Technology

The Interactive High Performance Computing Laboratory (IHPCL) is formed from an interdisciplinary group of computational scientists out of many different departments within the Georgia Institute of Technology. The primary interest of the group is in high performance scientific computing, with an emphasis on the real-time visualization and interaction with large data sets. The ability to enable remote collaboration in addition to remote visualization is a current strong motivating factor.

One archetype of the sort of problem we have been working on is a toolkit to interact with molecular dynamics (MD) packages. Molecular dynamics programs are used in many different disciplines to provide high-quality, atomic level simulations of materials. Some particular examples are studies of stress and wear in materials engineering, formation of nano-clusters and nano-jets in physics, studies of surface adsorption and catalysis in chemistry, and combustion in aerospace engineering. Despite the varied behaviors and subjects, most of the MD programs share certain common computational structures that can be used to create a common visualization service.

One immediate issue is that almost all of these programs are explicitly parallel. So some measures must be taken to interface with the parallel data structures. Fortunately, most such programs already have some facility for gathering data, and the visualization programs can tie into these existing subroutines. Another issue is the diversity of the languages used to write the packages – Fortran is prevalent on the computational end, while C and Java are preferred on the visualization end. As we shall see, this problem is resolved as a side benefit of our communications infrastructure.

Currently, the toolkit include several storage and visualization options, including direct storage of the binary data to disk, export to a parallel ray-tracer program (a parallel version of POVRay, called pvmpov), or VRML visualization. In addition, an option for turning the ray-traced images into a QuickTime movie is available. One of the key things lacking at this point is the ability to interact with other scientists and users while viewing all of these outputs. This is where we believe interacting with the Access Grid community can serve to help both interests.

One of the key pieces of technology which we have to offer is the methodology for data communications [1-4]. Our communications library, called ECho (for Event Channels), supports high bandwidth, platform independent event services for both data and control. It also has high language interoperability, with bindings in C and Java already completed, and the Fortran binding mostly finished. The platform independence and the interoperability are particularly relevant for the HPC to AG communications route, since most HPC codes run on Unix machines and the Access Grid display is a Java/Windows environment (at least in the current implementation).

In bandwidth comparisons to XML or Corba, ECho provides a significant advantage[3]. It also has superior latency performance. One interesting feature (of less immediate importance), is that the ECho library can multiplex its event dispersal over several different connections, including TCP, UDP, multicast, or wireless protocols. Although the Access Grid's current focus is mainly on multicast technology, some of the other protocols may be of use for streaming limited data out to local "docked" clients. Also, the robust connection management underlying the event services allows for an easy upgrade path as connection technologies change, while still providing backwards compatibility.

The new feature of ECho which also has great impact on our vision for the HPC-Access Grid union is what we call derived event channels. The **user**, at time of subscription to the event channel, may pass a function to the sources of the channel. Binary code for this function is dynamically generated and inlined. A new event channel is then derived from the previous content by the **source** applying the function as a filter to all data before sending something out. This leaves two channels; the first is the same as before, and the second has the filtered content of the first. This allows for intelligent bandwidth management, as well as careful control of things like formatting, data relevancy, and data transformation.

These functions can also chained, so that the final data stream leaving the source may be the parsed through multiple functions and filters. At each stage, other clients may be serviced from the derived channel before the data is filtered through the next function. Also, using new functionality in the connection manager, some of the functions may even be exported to a thirdparty machine. This is all done transparently from the user's end, but it allows for better load-balancing on the high-performance computing end.

Therefore, our view for a strongly coupled implementation of scientific visualization with the Access Grid uses this functionality to intelligently manage data as it streams from source to its multiple clients. As an AG node subscribes to the data stream, it may choose to register for stereographic modeling, or it may request an MPEG stream or the straight data. This also means that the client can manage its own bandwidth, by selecting data streams (or requesting transformed data streams) that match the available network performance. This also provides the opportunity for a collaborator to gather in data, perform additional manipulation (such as detection of crystal formation), and republish the transformed data for the community of collaborators.

Over top of this, of course, has to sit the collaboration technology of the Access Grid. There still remains a lot of work in studying both the individual implementation of the collaboration tools and the overall picture of how to make people **want** to use them. We hope that this research and the ECho infrastructure can serve as a part of the effort to push collaboration technology out to a broader community.

References

[1] The ECho Event Delivery System (User's Guide), Greg Eisenhauer, http://www.cc.gatech.edu/systems/projects/ECho

[2] A Middleware Toolkit for Client-Initiated Service Specialization, Greg Eisenhauer, Fabian Bustamente and Karsten Schwan, Proceedings of the PODC Middleware Symposium - July 18-20, 2000

[3] Event Services for High Performance Computing, Greg Eisenhauer, Fabian Bustamente and Karsten Schwan, Proceedings of High Performance Distributed Computing (HPDC-2000)

[4] JECho - Supporting Distributed High Performance Applications with Java Event Channels, Dong Zhou, Karsten Schwan, Greg Eisenhauer and Yuan Chen, To appear at Cluster2000

Downloadable versions of the above can be found at
http://www.cc.gatech.edu/systems/projects/ECho